(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 780 773 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
25.06.1997 Bulletin 1997/26

(51) Int Cl.[6]: **G06F 13/40**

(21) Application number: **96309078.2**

(22) Date of filing: **12.12.1996**

(84) Designated Contracting States:
DE FR GB

(30) Priority: **19.12.1995 US 580000**

(71) Applicant: **NCR International, Inc.**
Dayton, Ohio 45479 (US)

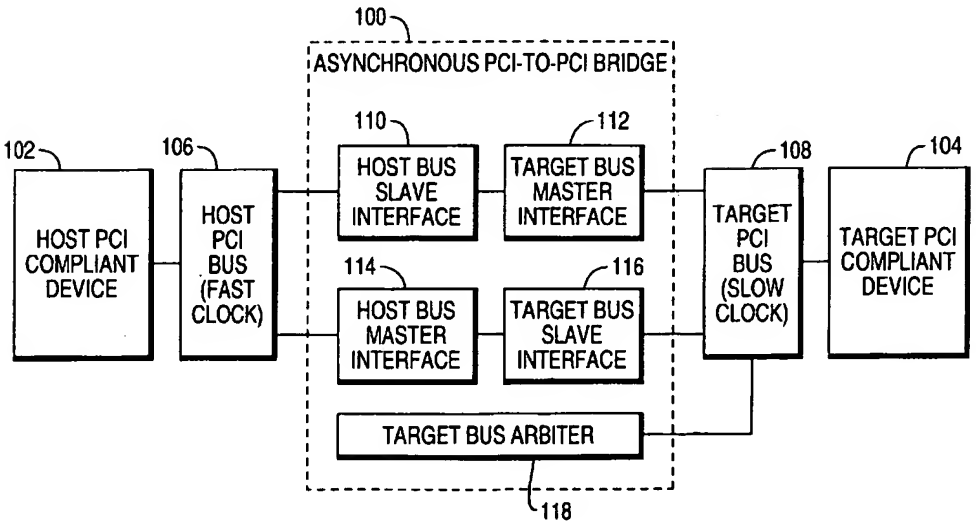
(72) Inventor: **Vo, Tri Tinh**
San Diego, CA 92129 (US)

(74) Representative: **Robinson, Robert George**
International Intellectual Property Department,
NCR Limited,
206 Marylebone Road
London NW1 6LY (GB)

(54) Asynchronous bus bridge

(57) An asynchronous bridge (100, 120) is provided to allow interoperability between devices (102, 104) operating at different clock frequencies. A first command message from the first device (102, 104) is received and stored in a bridge (100, 120). The first command message is transmitted to the second device (104, 102) and a reply message responsive to the first command message is received in the bridge (100, 120) and transmitted to the first device (102, 104). This asynchronous bridge is useful in the development of bus components and expansion boards and provides an interface between devices operating at independent or independently controllable clock speeds.

FIG. 1A



EP 0 780 773 A1

Description

The present invention relates generally to bus interface devices, and in particular, to a method and apparatus for bridging two devices operating at different clock frequencies.

The PCI local bus is a high performance 32- or 64-bit bus with multiplexed address and data lines. The PCI local bus was defined to establish an industry standard for a high performance local bus architecture offering low cost. The PCI component and add-in card interface is processor-independent, enabling an efficient transition to future processor generations, and may be used with multiple processor architectures. This processor independence allows the PCI local bus to be optimized for I/O functions, enabling concurrent operations of the local bus with the processor/memory subsystem, and accommodating multiple high performance peripherals in addition to graphics such as motion video, SCSI, and hard disk drives.

The movement to enhance the video and multi-media displays of high definition TV and other three-dimensional or high bandwidth I/O will continue to increase local bus bandwidth requirements. The PCI interface is particularly useful in interconnecting highly integrated peripheral controller components, peripheral add-in boards, and processor/memory systems. The PCI interface standard also offers additional benefits to users of PCI-based systems. Configuration registers are specified for PCI components and add-in cards. A system with embedded auto-configuration software, it offers true ease of use for the system user by automatically configuring PCI add-in cards at power-on. The PCI Local Bus Specification, Rev. 2.0, which is incorporated by reference herein, defines the PCI interface protocol, electrical, mechanical, and configuration specifications for PCI local bus components and expansion boards.

Situations arise whereby interoperability between bus components or devices with independent or independently controllable clocks is desired. A user may desire interoperability between a host computer and a peripheral operating at a slower or faster clock speed. Further, the clock speed of either the host computer, the peripheral, or both may be controllable or variable. For example, the development of bus components and expansion boards is expedited using hardware emulators. These emulators allow the developer to use easily-modified software to test the logic implemented on the expansion boards and verify the real-time functionality of the proposed design. However, the functional adaptability of the hardware emulators is not achieved without cost. Typically, these hardware emulators must run at much slower clock speeds than the host device, and are thus incapable of interfacing with the host bus at full speed. This problem is especially critical when the host bus serves a PCI-compliant device, because of the high I/O speed capability of the PCI bus.

It is an object of the present invention to provide a method and apparatus for bridging a first device operating at a first clock frequency with a second device operating at a second clock frequency.

According to one aspect of the invention there is provided a method of bridging a first device operating at a first clock frequency with a second device operating at a second clock frequency, characterized in that it comprises the steps of:

receiving a first command message from the first device in a bridge coupled between the first device and the second device, the message comprising a device command;
storing the first command message in the bridge;
transmitting the first command message from the bridge to the second device;
receiving a reply message responsive to the first command message from the second device in the bridge; and
transmitting the reply message to the first device.

According to another aspect of the invention there is provided an apparatus for bridging a first PCI-compliant device operating at a first clock frequency with a second PCI-compliant device operating at a second clock frequency, characterized in that it comprises a bridge coupled between the first PCI-compliant device and the second PCI compliant device, the bridge further comprising:

a first bus slave interface coupled to the first PCI-compliant device, the first bus slave interface comprising means for receiving a first command message, for storing the first command message, for commanding the first PCI-compliant device to re-transmit the command message, for receiving the re-transmitted command message from the first PCI-compliant device, for determining if the stored first command message matches the re-transmitted command message, for determining if a reply message responsive to the first command message is received, for transmitting the reply message to the first PCI-compliant device; and
a second bus master interface coupled to the first bus slave interface and the second PCI-compliant device, the second bus master interface further comprising means for transmitting the first command message to the second PCI-compliant device, and means for receiving the reply message from the second PCI-compliant device and forwarding the reply message to the first bus slave interface.

Such a method and apparatus provide an asynchronous bridge which is useful in the development of bus components and expansion boards, and which allows interoperability between bus components or devices by providing an interface between any devices operating

at independent or independently controllable clock speeds.

Embodiments of the invention will now be described by way of example with reference to the accompanying drawings, in which:

FIG. 1a presents a top-level block diagram of the present invention;

FIG. 1b presents a top-level block diagram of the present invention implemented on a computer;

FIG. 2 presents a flow chart of the operations performed by the host bus slave interface of the present invention;

FIG. 3 presents a flow chart of the operations performed by the target bus master interface of the present invention;

FIG. 4 presents a flow chart of the operations performed by the target bus slave interface;

FIG. 5 presents a flow chart of the operations performed by the host bus master interface; and

FIG. 6 presents a flow chart of the operations performed by the target bus arbiter.

FIG. 1a presents a top-level block diagram of the asynchronous PCI-to-PCI bridge (APB) (100). The bridge is coupled between a host PCI-compliant device (102) and a target PCI-compliant device (104) by a host PCI bus (106) and a target PCI bus (108) respectively. The APB (100) comprises a host bus slave interface (110) coupled to a target master bus interface (112), and a host bus master interface (114) coupled to a target bus slave interface (116). A target bus arbiter (118) is coupled to the target bus. The function and operation of each of these components is further discussed below.

It is noted that the present invention need not be limited in application or scope to PCI-compliant devices. Instead, the method and apparatus described herein can serve as an asynchronous bridge between any devices complying with a predefined interface protocol. Further, many possible implementations of bridge illustrated in FIG. 1a are possible. One possible implementation would utilize commonly available flip flops and other logical memory devices to create a special purpose APB (10). Similarly, the functions described herein performed by the APB (100) could be performed by a microprocessor coupled to a memory device. Finally, the elements shown in FIG. 1a could be implemented by a computer (120) coupled between the host PCI bus (106) and the target PCI bus (108), as shown in FIG. 1b. The computer may include, inter alia, a processor (122), a memory (124), as well as a fixed and/or removable storage device (126) and associated media (128). Those skilled in the art will recognize that any combination of the above components, or any number of different components, peripherals, and other devices may be used to implement the processes and structures defined herein.

FIG. 2 presents a flow chart of the operations performed by the host bus slave interface (110). The process begins and ultimately ends with the host PCI bus (106) in an idle state (200). From the idle state (200), the host bus slave interface (110) determines whether the host bus has requested access (202) to the bridge. This is indicated when the FRAME signal pin on the host PCI bus (106) is asserted active. This is indicated when the FRAME pin is logically low. Next, the instruction from the host PCI bus (106) is examined to determine if the instruction is an instruction supported (204) by the target PCI device (104) and the APB (100). For example, invalidate instructions only apply to the system memory of the host motherboard, and are not supported by the APB or the target PCI device. If this instruction was indicated, the APB returns to the idle state (200). Thereafter, the host slave bus interface (110) determines (206) if the APB was selected, using the message from the host PCI-compliant device (102). Ordinarily, this is determined from the address or address range or window as specified by the message. If the address is within the APB (100) address window, the address, data, and command are captured (208). This involves storing these values to save them for later use. This data will later be overwritten in subsequent trials, as described below.

Next, the host slave bus interface (110) determines whether this was the first trial for the captured command (208). If this is not the first trial for this command or instruction, the instruction address, command, and data is stored for later use. This stored data is later used to implement a delayed action interface between the two PCI devices. After storing (212) the instruction, the host slave bus interface (110) sets (214) a HOST_BUSY flag and a 1ST_TRIAL flag. The HOST_BUSY flag indicates that the host PCI-compliant device (102) is accessing the APB (100). This flag is later used to notify other elements of the APB that the host is busy. The 1ST_TRIAL flag is used to implement the function indicated in block (210).

Next, after receiving the command instruction from the host PCI bus (106), the target bus master interface (112) is accessed (216), and further processing (218) is performed. Since clock frequency and hence the speed of the target PCI bus (108) may be lower than the host PCI bus (106), it is undesirable to hold the host PCI bus (106). To avoid holding the host PCI bus (106), retry handshakes are asserted (220) to the host PCI bus (106). This frees the host PCI bus (106) for other operations while data is retrieved from or written to the target PCI bus (108). Retry handshake assertion (220) is performed in compliance with PCI interface protocols which are well known in the art, and described in the PCI Local Bus Specification, Rev. 2.0, incorporated by reference herein. In one embodiment of the present invention, a delay is inserted before retry handshakes are asserted (220) to the host PCI Bus (106). This delay allows other components to access the PCI Bus (106). This time delay can be set to any desired value, but should be at

least two clock periods.

While the retry handshakes are being asserted (220) to the host PCI (106), the target bus master interface (112) is activated (216) by generating a HOSTREQ signal, and passing that signal to the target bus master interface (112).

FIG. 3 is a flow chart that describes the functions performed by the bus master interface (112). From the idle state (200), the target bus master interface (112) checks to see if access to the target bus has been requested (302), by examining the HOSTREQ signal. If target bus access has not been requested, the target bus master interface (112) returns to the idle state (200). If target bus access has been requested, the target bus is queried to determine if it is busy (304). This is accomplished by examining a QBUSY signal generated by the target bus arbiter (118), as described in detail later in this specification. If the target bus is busy, it is retried until the bus is no longer busy. When the target bus is not busy, access handshakes are asserted (306) to the target bus. This procedure is well known in the art, and described in detail in the PCI Local Bus Specification, Rev. 2.0, incorporated by reference herein.

Thereafter, the instruction or command information that was stored (212) in the APB (100) is outputted to the target bus master interface (112). This data is later asserted to the target PCI bus (108). Next, the target bus master interface (112) checks to determine if the access process is completed (308) indicating that the target PCI device (104) is ready to accept the instruction or command. If the target PCI device (104) is not prepared to accept the instruction or command (310), the access process begins anew. If the target PCI device (104) and the target PCI bus (108) is prepared to accept the command, processing continues. If an abort command was received (312) from the target PCI-compliant device (104), that target access was completed is reported (314) to the host slave bus interface (110), and abort handshakes are asserted to the target bus. This process is well known in the art, and described in detail in the PCI Local Bus Specification, Rev. 2.0, incorporated by reference herein.

If no abort command was received from the target PCI-compliant device (104), the stored instruction or command is asserted to the target PCI bus (108). If the instruction was a write command, the data is written (318) to the target PCI bus (108), and if the instruction was a read command, the data is read (318) from the target PCI bus (108). Thereafter, that target access was completed is reported (320) to the host slave bus interface (110), and normal completion handshakes are asserted (322) to the target bus.

Referring back to FIG. 2, the reader will recall that while the target bus master interface (112) was activated (216), retry handshakes were asserted (220) to the host PCI bus (106) to command the host bus to re-transmit the given instruction. Since the host PCI bus (106) is faster than the target PCI bus, this prevents the host PCI bus from being hung-up waiting for the target PCI bus to respond. In one embodiment, a time delay may be implemented before the retry handshakes are asserted to allow other devices to access the host PCI bus (106). After the host PCI bus (106) responds to the retry command with a re-transmitted command, the host slave bus interface (110) responds as described earlier by capturing (208) the re-transmitted command if the instruction is supported (204), and the APB is selected (206). Next, the HOST_BUSY flag is examined (210) to determine if this is the first pass through the APB (100) for this instruction.

Assuming the process previously described have taken place, the HOST_BUSY flag will be set, and the host slave bus interface (110) will then compare (222) the stored information from the first pass with the captured current instruction. If the stored information from the first pass and the captured current instruction do not match (224), retry handshakes are again asserted (220) to the host PCI bus (106), as described earlier. If the stored information and the captured current instructions match (224) and target bus access is completed (226), the host slave bus interface checks to see if an abort command was received (228) from the target PCI-compliant device (104).

If so, the HOST_BUSY and 1ST_TRIAL flags are cleared (230), and abort handshakes are asserted (232) to the host PCI bus (106). If no abort command was received from the target PCI-compliant device (104), processing continues as shown in blocks (234-236). If the command instruction was to read data from the target PCI-compliant device (104), this data is returned (234) from the target bus master interface (112) to the host PCI bus (106). Finally, processing is completed by clearing (236) the HOST_BUSY flag and the 1st_TRIAL flag and asserting normal completion handshakes (238) to the host PCI bus (106). This completes the process wherein the host PCI-compliant device (102) performs as the master, and the target PCI-compliant device (104) performs as the slave.

Figures 4 and 5 are flow charts illustrating the operation of one embodiment of the present invention whereby the slower clocked PCI-compliant device operates as the master and the faster clocked PCI-compliant device operates as the slave.

FIG. 4 illustrates the operations performed by the target bus slave interface (116). As before, the process begins from the idle state (200). First, the target bus slave interface (116) determines (402) if the target bus requests access to the host PCI bus (106) by examining the QREQ flag. Next, the address, data, and command of the instruction from the target PCI bus are captured (404). This involves storing these values for later use. Next, the Q_BUSY flag is set, and the host bus master interface is activated. The processes associated with the host bus master interface (114) are described below.

FIG. 5 is a flow chart describing the operations performed by the host bus master interface (114). From the

idle state (200), the host bus master interface (114) determines if access to the host bus was requested (502) by examining the QREQ flag. If access was not requested, the host bus master interface (114) returns to the idle state (200). If access was requested, the host bus master interface (114) requests access to the host bus by activating a H_REQ signal. While the host PCI bus may grant the bus (as indicated by the GNT signal), the host PCI bus may not be idle. Instead, the GNT signal merely indicates that the requestor will be the next owner of the bus. By assuring that the host PCI bus (106) is granted and idle before continuing, the host PCI bus is not unduly slowed down waiting for a target PCI bus (108) response. Further, by obtaining the bus grant, the host bus master interface (114) reserves the bus for use as soon as it becomes idle.

The host bus master interface (114) waits until the host PCI bus is both granted and idle. When this occurs, access handshakes with the host bus are begun and the information captured from the target bus (404) is output to the host bus master interface (114.) In one embodiment of the invention, to further assure proper communications, access handshakes are delayed until another GNT signal is received from the host PCI bus (106). These access handshake protocol are well known in the art and are described in the PCI Local Bus Specification, Rev. 2.0, incorporated by reference herein. Next, the host bus master interface (114) determines if access to the host PCI bus (106) is completed (510). After access is completed, the host master bus interface (114) determines if a retry command was received from the host PCI bus (106). If a retry command was received (512), retry handshakes are asserted to the host bus (514), using the protocol defined in the PCI Local Bus Specification referenced herein. If a retry command was not received, the host bus master interface checks to determine if a host abort command was received (516). If a host abort command was received, a H_DONE flag is set to report (518) a message to the target bus slave interface (116) that host PCI bus access was successfully completed. Thereafter, abort handshakes are asserted (520) to the host PCI bus (106). If an abort command was not received, the host bus master interface (114) performs the command or instruction supplied from the target bus slave interface (116). If the command was a read command, the data from the host PCI bus (106) is read (522). If the command was a write command, the indicated data is written (522) to the host PCI bus (106). Thereafter, a H_DONE flag is set to report (518) a message to the target bus slave interface 116 that host PCI bus access was successfully completed, and normal completion handshakes are asserted to the host PCI bus (106). Of course, if an illegal or unsupported instruction was provided, the command will not be supplied to the host PCI bus (106). This error may be reported to the target PCI bus (108), or may simply cause the APB to hang up. Further, illegal instructions may be detected and reported in the target bus slave interface (116).

Returning to FIG. 4, while host bus master interface (114) processes are being performed, the target bus slave interface (116) checks to see if the host PCI bus access is completed (412), by examining an H_DONE signal. If so, the target bus slave interface (116) checks (414) to see if the host PCI bus has aborted. If so, the Q_BUSY flag is cleared (416), and abort handshakes are asserted (418) to the target PCI bus (108). If no abort message was received, any data read from the host PCI bus (106) is returned to the target PCI bus (108). Thereafter, the Q_BUSY flag is cleared (422), and normal completion handshakes are asserted (424) to the target PCI bus (108).

FIG. 6 presents a flow chart describing the functions performed by the target bus arbiter (118). From the idle state (200), the target bus arbiter (118) determines if access to the target bus has been requested. This is indicated by a QREQ signal. When the host PCI bus (106) is not busy (604) as indicated by the HBUSY signal, the target PCI bus (108) is granted (606) to the bus requestor by setting the QGRANT signal. Thereafter, the QBUSY signal is set, indicating that the target PCI bus (108) is busy. The target bus arbiter (118) then waits until the target PCI bus (108) begins access as indicated by the QFRAME signal, and then removes the QGRANT signal, indicating that the target bus is no longer granted to the requestor. Finally, after the target PCI bus access is completed (614), the QBUSY flag is cleared (616), indicating that the target PCI bus is no longer busy, and the target bus arbiter returns to the idle state (200), and waits for the next request.

It is noted that while the target PCI-compliant device (104) is described above as operating at a clock frequency lower than that of the host PCI-compliant device (102), the device clock frequencies need not be so related. The present invention provides a bridge between any devices with independent or independently controllable clocks. Hence, the present invention may be practiced with a target PCI compliant device (104) with a faster clock frequency than that of the host PCI-compliant device (102), or with a PCI compliant device (104) with a variable or controllable clock frequency.

Claims

1. A method of bridging a first device (102, 104) operating at a first clock frequency with a second device (104, 102) operating at a second clock frequency, characterized in that it comprises the steps of:

   receiving a first command message from the first device (102, 104) in a bridge (100, 120) coupled between the first device (102, 104) and the second device (104, 102), the message comprising a device command;
   storing the first command message in the

bridge (100, 120);
transmitting the first command message from the bridge (100, 120) to the second device (104, 102);
receiving a reply message responsive to the first command message from the second device (104, 102) in the bridge (100, 120); and
transmitting the reply message to the first device (102, 104).

2. The method of claim 1 characterized in that the first device (102, 104) is a PCI-compliant device, the second device (104, 102) is a PCI-compliant device, and the device command is a PCI-compliant device command.

3. The method of claims 1 or 2 characterized in that the first device (102) is a host device and the second device (104) is a target device.

4. The method of claims 1 or 2 characterized in that the first device (104) is a target device and the second device (102) is a host device.

5. The method of claim 3 or 4 characterized in that:

the command message comprises a write command and data or a read command; and
the reply message comprises a message indicating that the data was written to the second PCI-compliant device (104; 102) or data from the second PCI-compliant device (104, 102) responsive to the read command respectively.

6. The method of claim 3 characterized in that it further comprises the steps of:

(a) commanding the first device (102) to re-transmit the first command message;
(b) receiving the re-transmitted command message from the first device (102) in the bridge (100);
(c) repeating steps (a)-(b) until the stored first command message matches the re-transmitted message; and
(d) repeating steps (a)-(c) until the reply message is received from the second device (104),

7. The method of claim 6 characterized in that the method further comprises the step of delaying for a time period of at least two first clock periods before commanding the first PCI-compliant device (102) to re-transmit the command message.

8. The method of claim 6 characterized in that the step of receiving the first command message further comprises the steps of:

determining if the PCI-compliant device command is supported by the second PCI-compliant device (104); and
returning the bridge (100) to an idle state when the PCI-compliant device command is not supported by the second PCI-compliant device (104).

9. The method of claim 4 characterized in that it further comprises a step of accessing the second PCI-compliant device (102) comprises the steps of:

(a) requesting access to the second PCI-compliant device (102) by transmitting a request signal from the bridge (100) to the second PCI-compliant device (102);
(b) receiving a signal from the second PCI-compliant device (102) in the bridge (100) indicating that the second PCI-compliant device (102) has been granted to the bridge;
(c) receiving a signal indicating that the second PCI-compliant device (102) is idle;
(d) releasing the second PCI-compliant device (102) if the first PCI-compliant device (102) is not idle;
(e) repeating steps (a)-(d) until the second PCI-compliant device (102) is both granted and idle; and
(f) performing PCI access handshaking and transmitting the first command message from the bridge (100) to the second device (102) when the second PCI-compliant device (102) is both granted and idle.

10. The method of claim 9 characterized in that it further comprises the step of repeating the step of receiving a signal from the second PCI-compliant device (102) in the bridge (100) indicating that the second PCI-compliant device (102) has been granted to the bridge (100) before performing the step of performing PCI access handshaking.

11. An apparatus for bridging a first PCI-compliant device (102) operating at a first clock frequency with a second PCI-compliant device (104) operating at a second clock frequency, characterized in that it comprises a bridge (100) coupled between the first PCI-compliant device (102, 104) and the second PCI compliant device (102, 104), the bridge (100) further comprising:

a first bus slave interface (110) coupled to the first PCI-compliant device (102), the first bus slave interface (110) comprising means for receiving a first command message, for storing the first command message, for commanding the first PCI-compliant device (102) to re-transmit the command message, for receiving the re-

transmitted command message from the first PCI-compliant device (102), for determining if the stored first command message matches the re-transmitted command message, for determining if a reply message responsive to the first command message is received, for transmitting the reply message to the first PCI-compliant device (102); and

a second bus master interface (112) coupled to the first bus slave interface (110) and the second PCI-compliant device (104), the second bus master interface (112) further comprising means for transmitting the first command message to the second PCI-compliant device (104), and means for receiving the reply message from the second PCI-compliant device (104) and forwarding the reply message to the first bus slave interface (110).

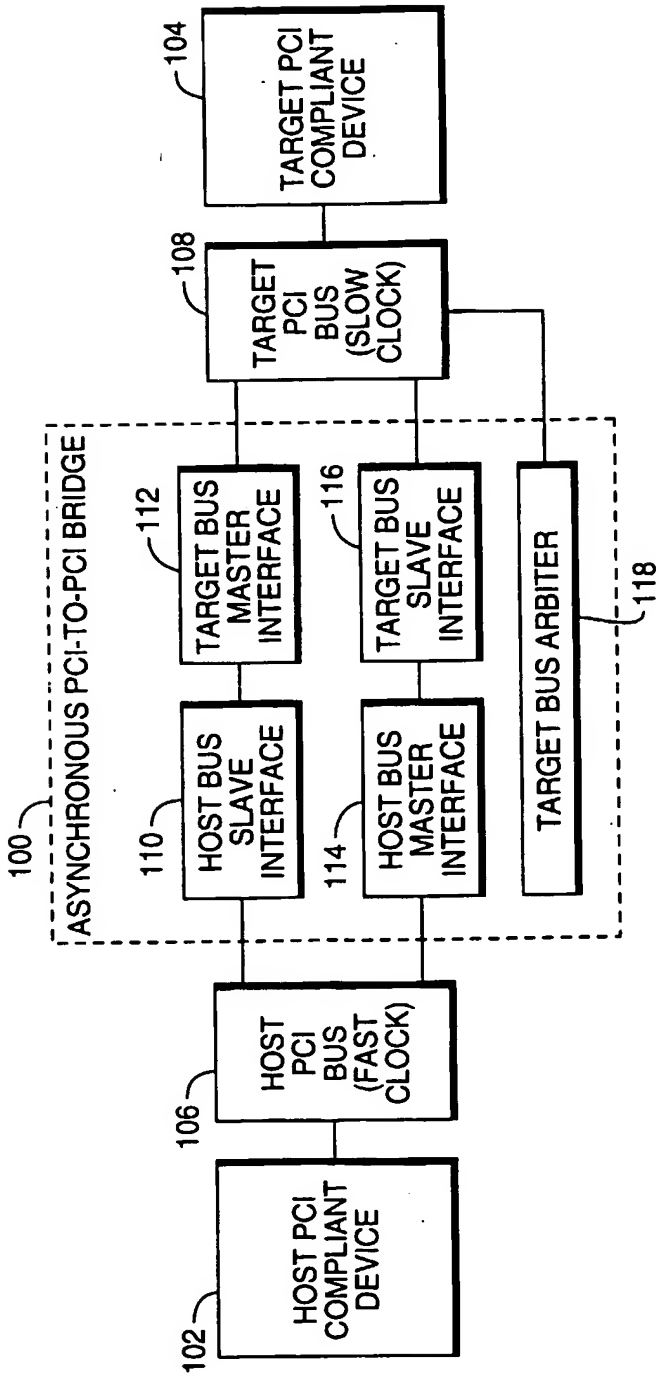
12. The apparatus of claim 11 characterized in that the bridge (100) further comprises a target bus arbiter (118) for arbitrating access to the second PCI-compliant device (104).

13. The apparatus of claim 11 characterized in that the bridge (100) further comprises:

a second bus slave interface (116) coupled to the second PCI-compliant device (104), the second bus slave interface (116) comprising means for receiving a first command message from the second PCI-compliant device (104), the message comprising a PCI-compliant device command, means for storing the first command message, means for transmitting a reply message responsive to the first command message to the second PCI-compliant device (104); and

a first bus master interface (114) coupled to the second bus slave interface (116) and the first PCI-compliant device (102), comprising means for accessing to the first PCI-compliant device (102), means for transmitting the first command message to the first PCI-compliant device (102) when the first PCI-compliant device (102) is available and means for receiving the reply message from the first PCI-compliant device (102).

FIG. 1A


HOST PCI COMPLIANT DEVICE
102
HOST PCI BUS (FAST CLOCK)
106
HOST BUS SLAVE INTERFACE
110
HOST BUS MASTER INTERFACE
114
ASYNCHRONOUS PCI-TO-PCI BRIDGE
100
TARGET BUS MASTER INTERFACE
112
TARGET BUS SLAVE INTERFACE
116
TARGET BUS ARBITER
118
TARGET PCI BUS (SLOW CLOCK)
108
TARGET PCI COMPLIANT DEVICE
104

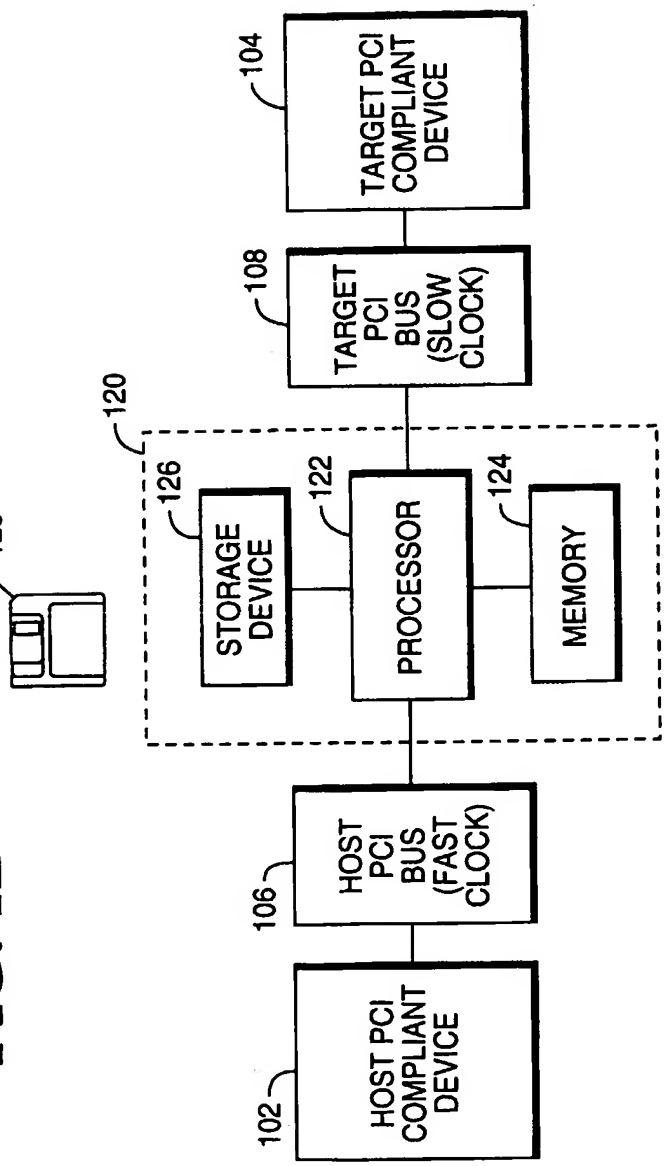

102
HOST PCI COMPLIANT DEVICE
106
HOST PCI BUS (FAST CLOCK)
120
126
STORAGE DEVICE
122
PROCESSOR
124
MEMORY
128
108
TARGET PCI BUS (SLOW CLOCK)
104
TARGET PCI COMPLIANT DEVICE

FIG. 1B

FIG. 2A

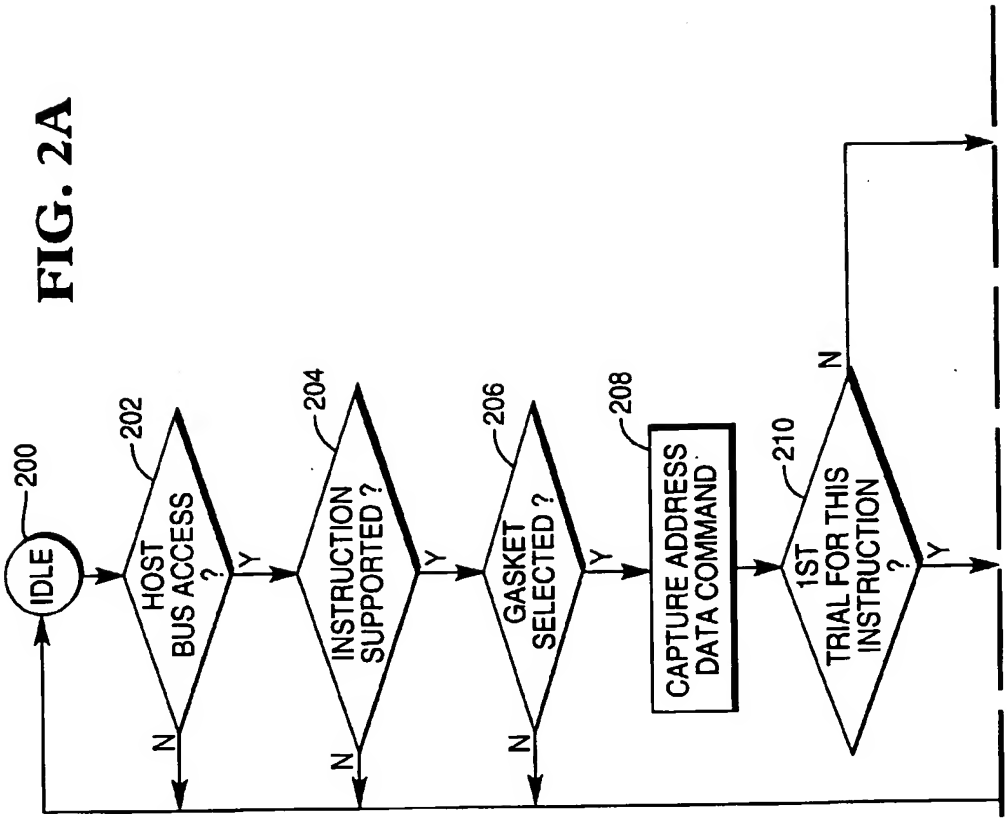

200
IDLE
202
HOST BUS ACCESS ?
Y
N
204
INSTRUCTION SUPPORTED ?
Y
N
206
GASKET SELECTED ?
Y
N
208
CAPTURE ADDRESS DATA COMMAND
210
1ST TRIAL FOR THIS INSTRUCTION ?
Y
N

FIG. 2B

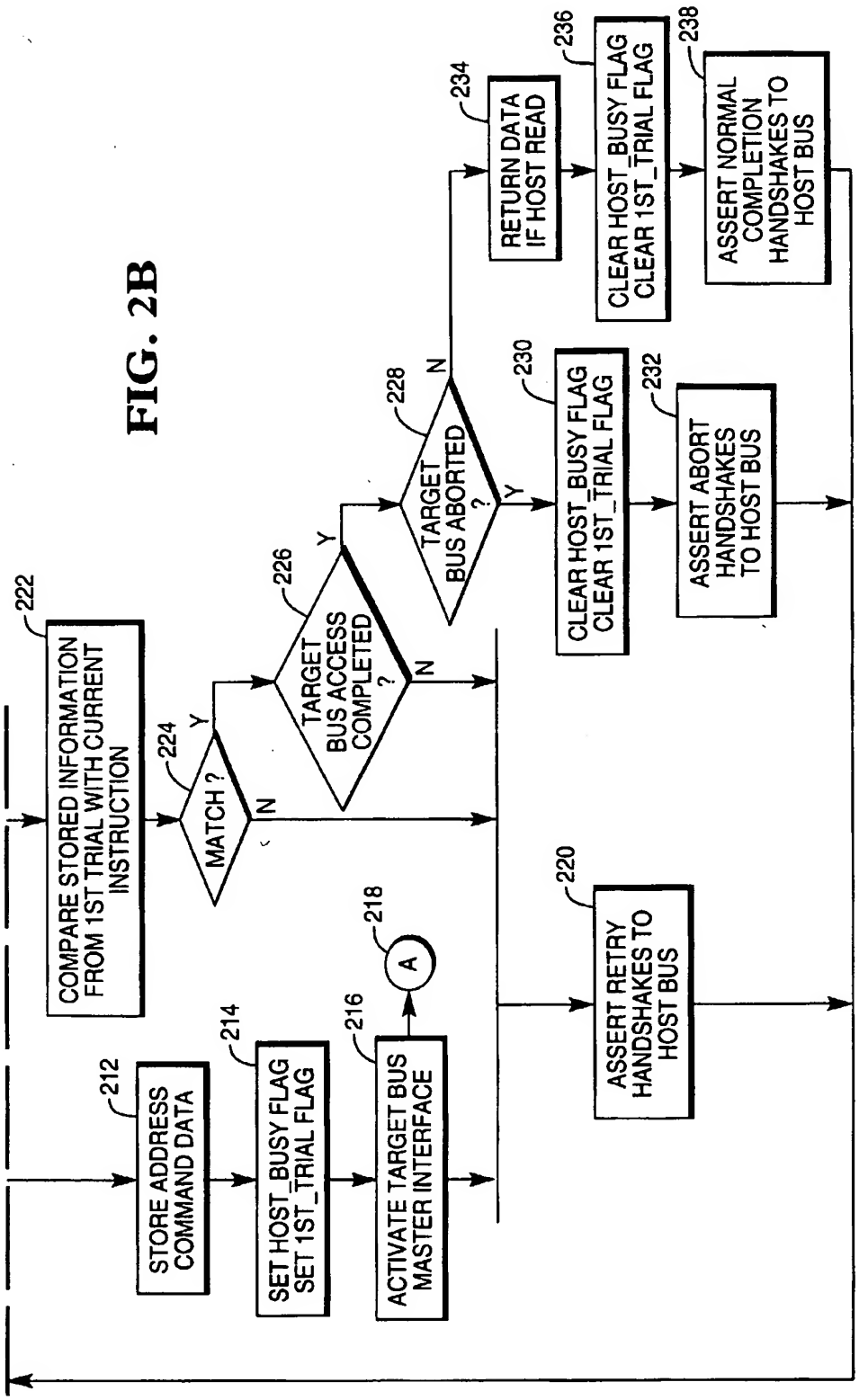

212
STORE ADDRESS COMMAND DATA
214
SET HOST_BUSY FLAG SET 1ST_TRIAL FLAG
216
ACTIVATE TARGET BUS MASTER INTERFACE
218
A
220
ASSERT RETRY HANDSHAKES TO HOST BUS
222
COMPARE STORED INFORMATION FROM 1ST TRIAL WITH CURRENT INSTRUCTION
224
MATCH ?
Y
N
226
TARGET BUS ACCESS COMPLETED ?
Y
N
228
TARGET BUS ABORTED ?
Y
N
230
CLEAR HOST_BUSY FLAG CLEAR 1ST_TRIAL FLAG
232
ASSERT ABORT HANDSHAKES TO HOST BUS
234
RETURN DATA IF HOST READ
236
CLEAR HOST_BUSY FLAG CLEAR 1ST_TRIAL FLAG
238
ASSERT NORMAL COMPLETION HANDSHAKES TO HOST BUS

FIG. 3

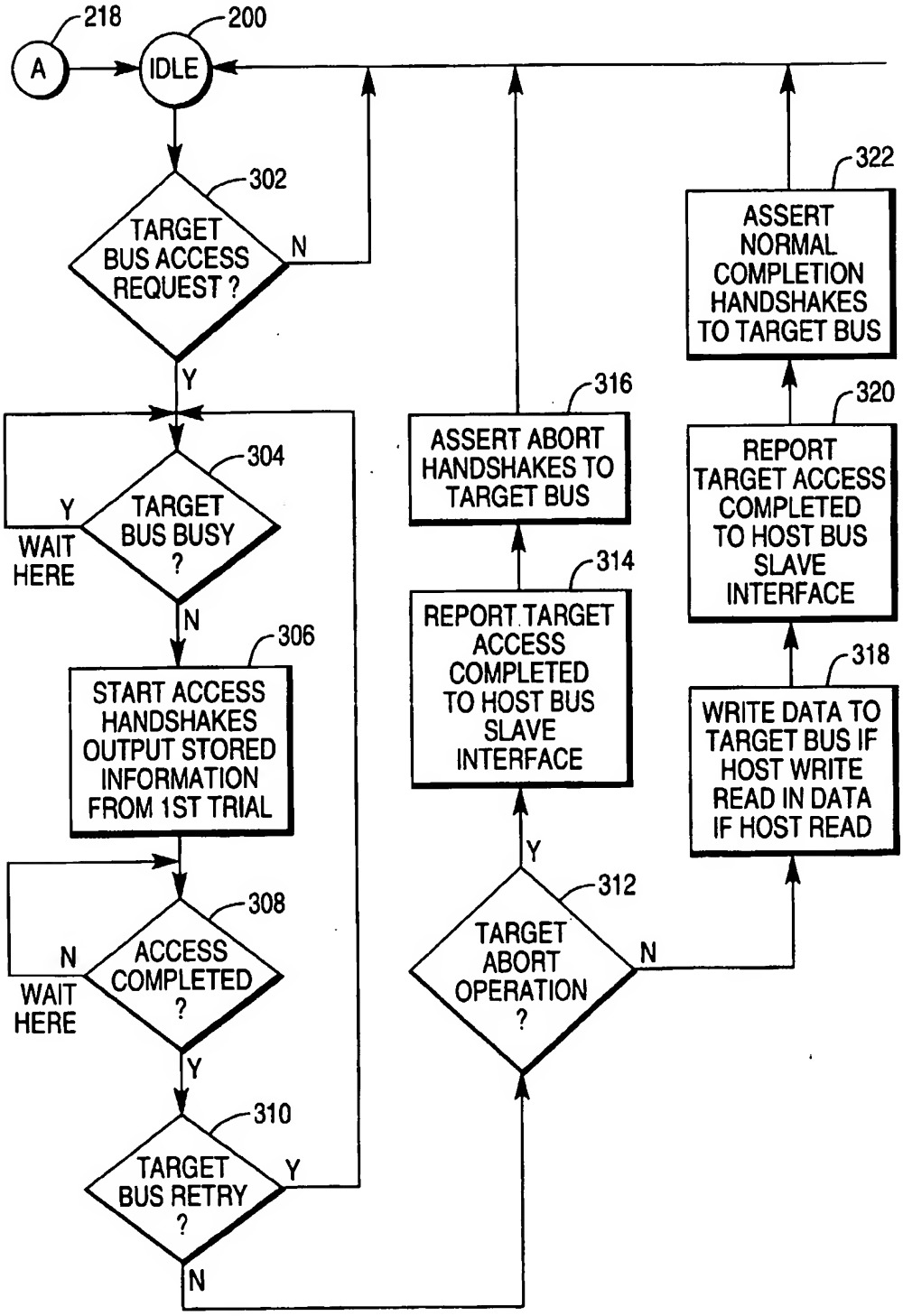

218
A
200
IDLE
302
TARGET BUS ACCESS REQUEST ?
N
Y
304
TARGET BUS BUSY ?
Y
WAIT HERE
N
306
START ACCESS HANDSHAKES OUTPUT STORED INFORMATION FROM 1ST TRIAL
308
ACCESS COMPLETED ?
N
WAIT HERE
Y
310
TARGET BUS RETRY ?
Y
N
316
ASSERT ABORT HANDSHAKES TO TARGET BUS
314
REPORT TARGET ACCESS COMPLETED TO HOST BUS SLAVE INTERFACE
Y
312
TARGET ABORT OPERATION ?
N
322
ASSERT NORMAL COMPLETION HANDSHAKES TO TARGET BUS
320
REPORT TARGET ACCESS COMPLETED TO HOST BUS SLAVE INTERFACE
318
WRITE DATA TO TARGET BUS IF HOST WRITE READ IN DATA IF HOST READ

FIG. 4

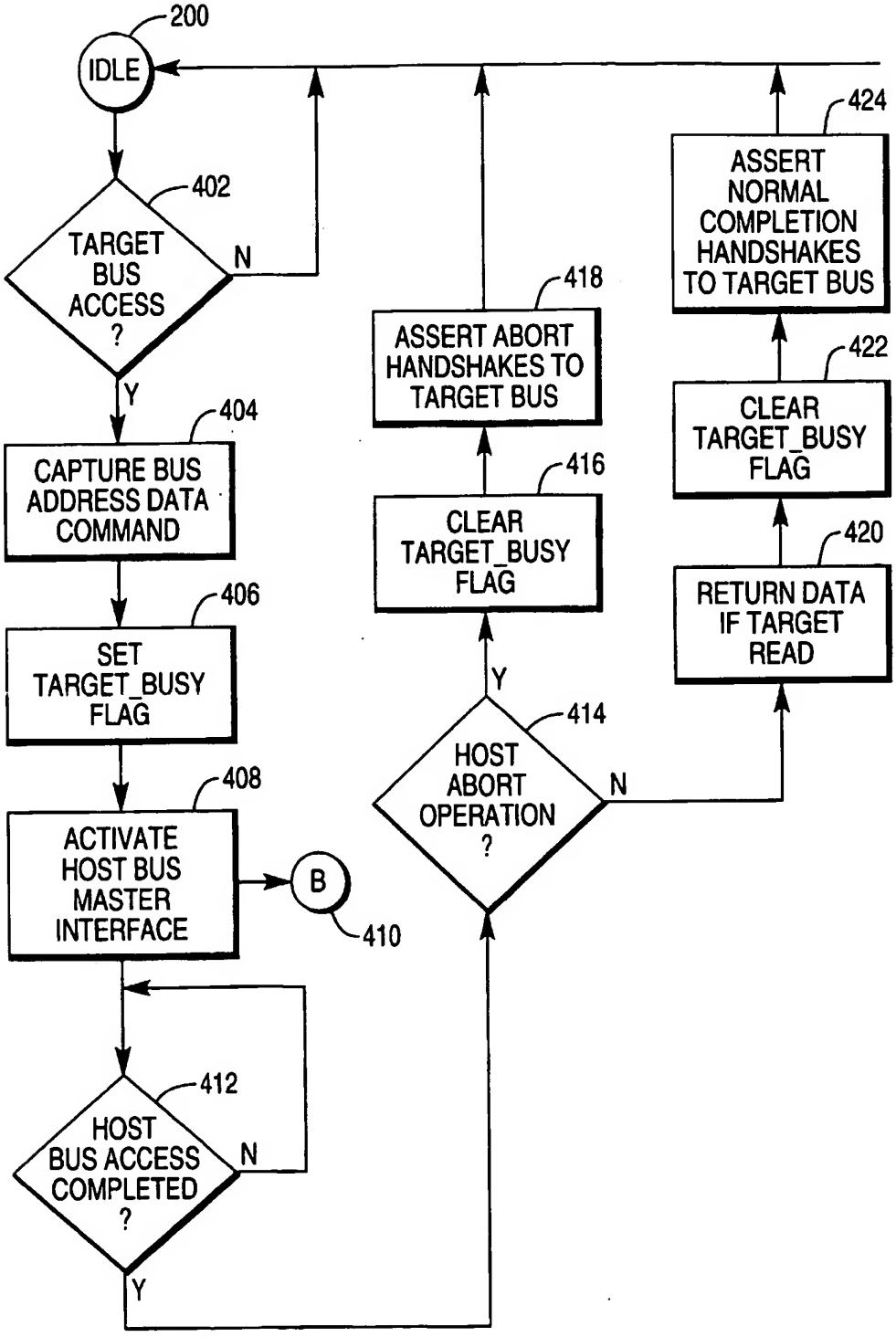

200
IDLE
402
TARGET BUS ACCESS ?
N
Y
404
CAPTURE BUS ADDRESS DATA COMMAND
406
SET TARGET_BUSY FLAG
408
ACTIVATE HOST BUS MASTER INTERFACE
B
410
412
HOST BUS ACCESS COMPLETED ?
N
Y
414
HOST ABORT OPERATION ?
Y
N
416
CLEAR TARGET_BUSY FLAG
418
ASSERT ABORT HANDSHAKES TO TARGET BUS
420
RETURN DATA IF TARGET READ
422
CLEAR TARGET_BUSY FLAG
424
ASSERT NORMAL COMPLETION HANDSHAKES TO TARGET BUS

FIG. 5A

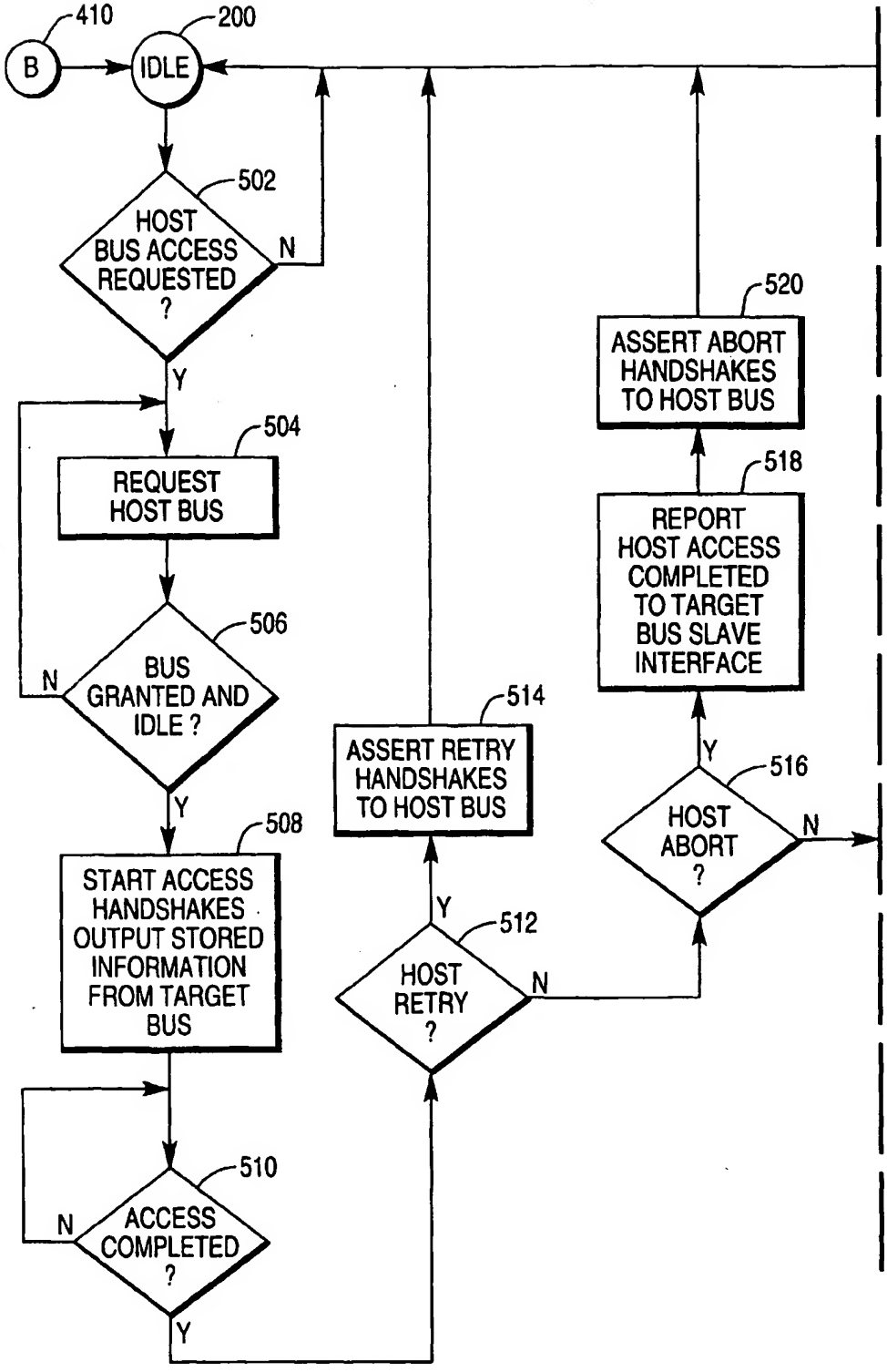

410
B
200
IDLE
502
HOST BUS ACCESS REQUESTED ?
N
Y
504
REQUEST HOST BUS
506
BUS GRANTED AND IDLE ?
N
Y
508
START ACCESS HANDSHAKES OUTPUT STORED INFORMATION FROM TARGET BUS
510
ACCESS COMPLETED ?
N
Y
514
ASSERT RETRY HANDSHAKES TO HOST BUS
512
HOST RETRY ?
Y
N
516
HOST ABORT ?
Y
N
518
REPORT HOST ACCESS COMPLETED TO TARGET BUS SLAVE INTERFACE
520
ASSERT ABORT HANDSHAKES TO HOST BUS

FIG. 5B

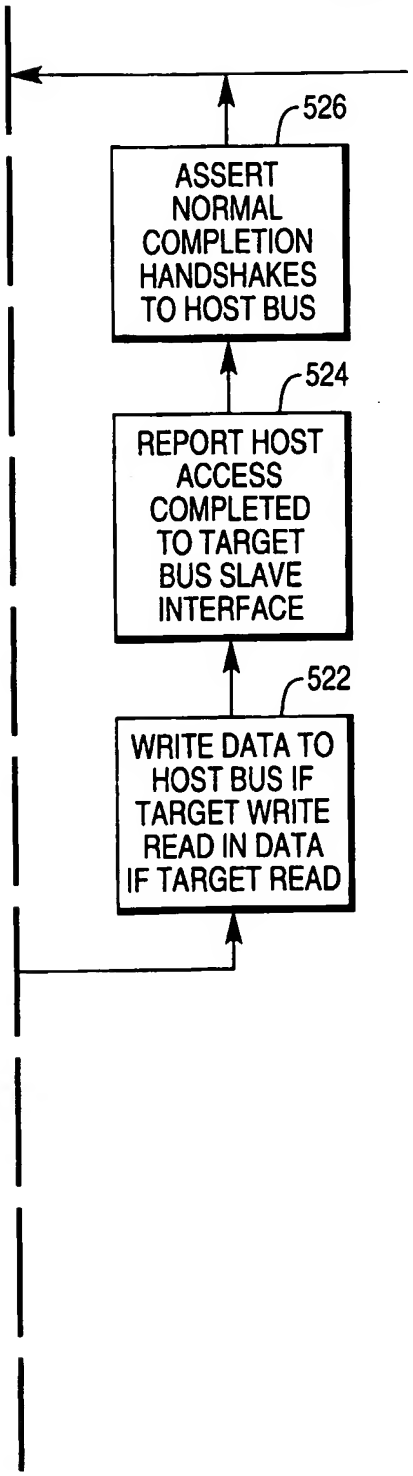
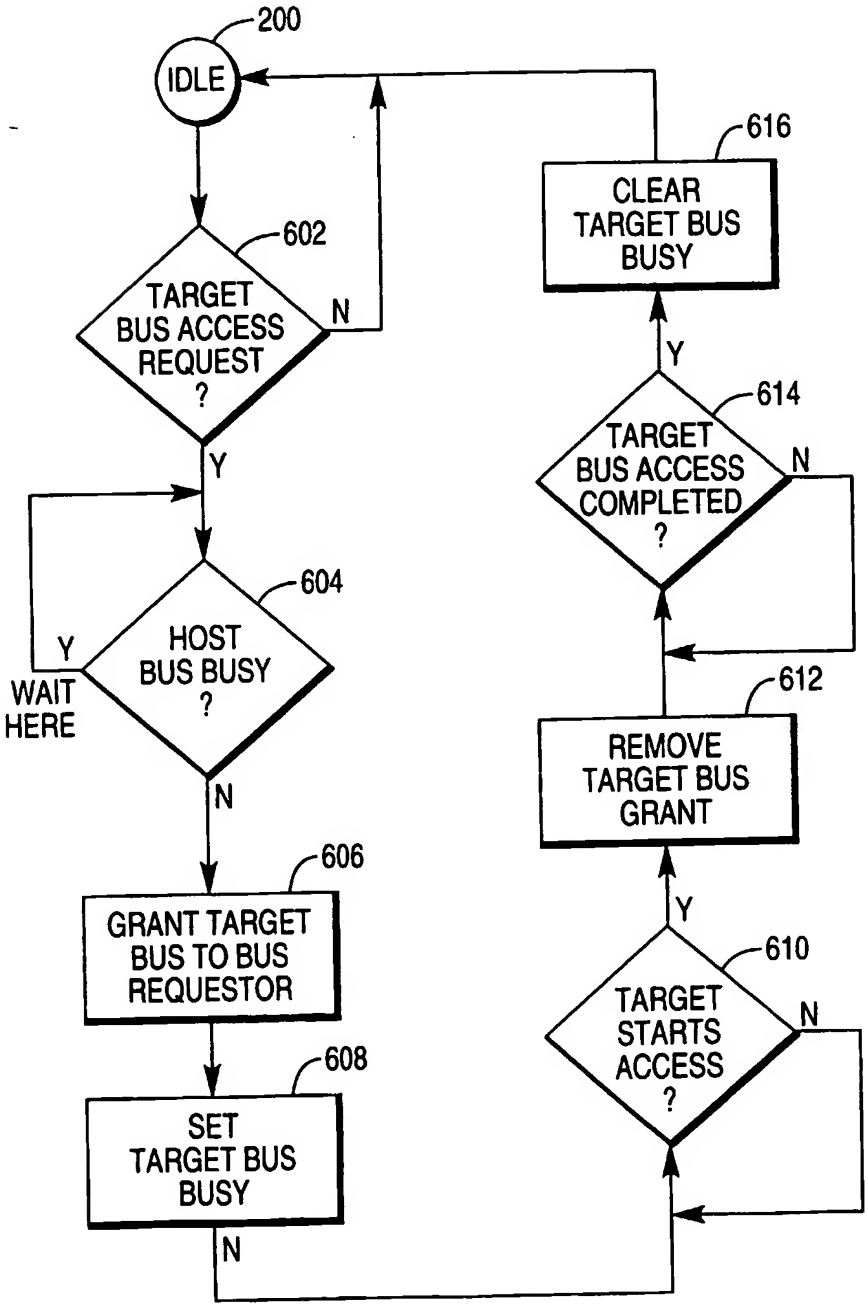

526
ASSERT NORMAL COMPLETION HANDSHAKES TO HOST BUS
524
REPORT HOST ACCESS COMPLETED TO TARGET BUS SLAVE INTERFACE
522
WRITE DATA TO HOST BUS IF TARGET WRITE READ IN DATA IF TARGET READ

FIG. 6


200
IDLE
602
TARGET BUS ACCESS REQUEST ?
N
Y
604
HOST BUS BUSY ?
Y
WAIT HERE
N
606
GRANT TARGET BUS TO BUS REQUESTOR
608
SET TARGET BUS BUSY
N
610
TARGET STARTS ACCESS ?
N
Y
612
REMOVE TARGET BUS GRANT
614
TARGET BUS ACCESS COMPLETED ?
N
Y
616
CLEAR TARGET BUS BUSY

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 96 30 9078

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
X	EP 0 645 715 A (IBM CORPORATION) * column 2, line 3 - column 3, line 15 * * column 6, line 4 - line 50 * * claims 1-5; figure 2 *	1,3,4	G06F13/40
A	EP 0 384 621 A (DATA GENERAL CORPORATION) * column 2, line 6 - column 3, line 8 * * column 4, line 25 - line 52 * * claims 1-4; figures 1,2 *	1-13	
A	US 5 455 915 A (COKE) * column 2, line 15 - line 36 * * column 4, line 32 - column 5, line 9 * * claims 1-4,6; figures 1,2 *	1-13	
			TECHNICAL FIELDS SEARCHED (Int.Cl.6)
			G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	2 April 1997	McDonagh, F

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)